VNE 1
VN: 10.10.0.1
VM:255.255.255.255

110

112

VN
VN:10.10.2.0
VM:255.255.255.0

VN: Virtual Subnet
VM: Virtual Netmask

150-1

150-2

• • •

150-n

WEB SITE 1
10.10.0.1

WEB SITE 2
10.10.2.1

APP 2
10.10.2.2

Computer 1

Computer 2

Computer 3

Global Virtual Address Space:
10.10.0.0

106

FIREWALL

-104

100

INTERNET

Remote User 1

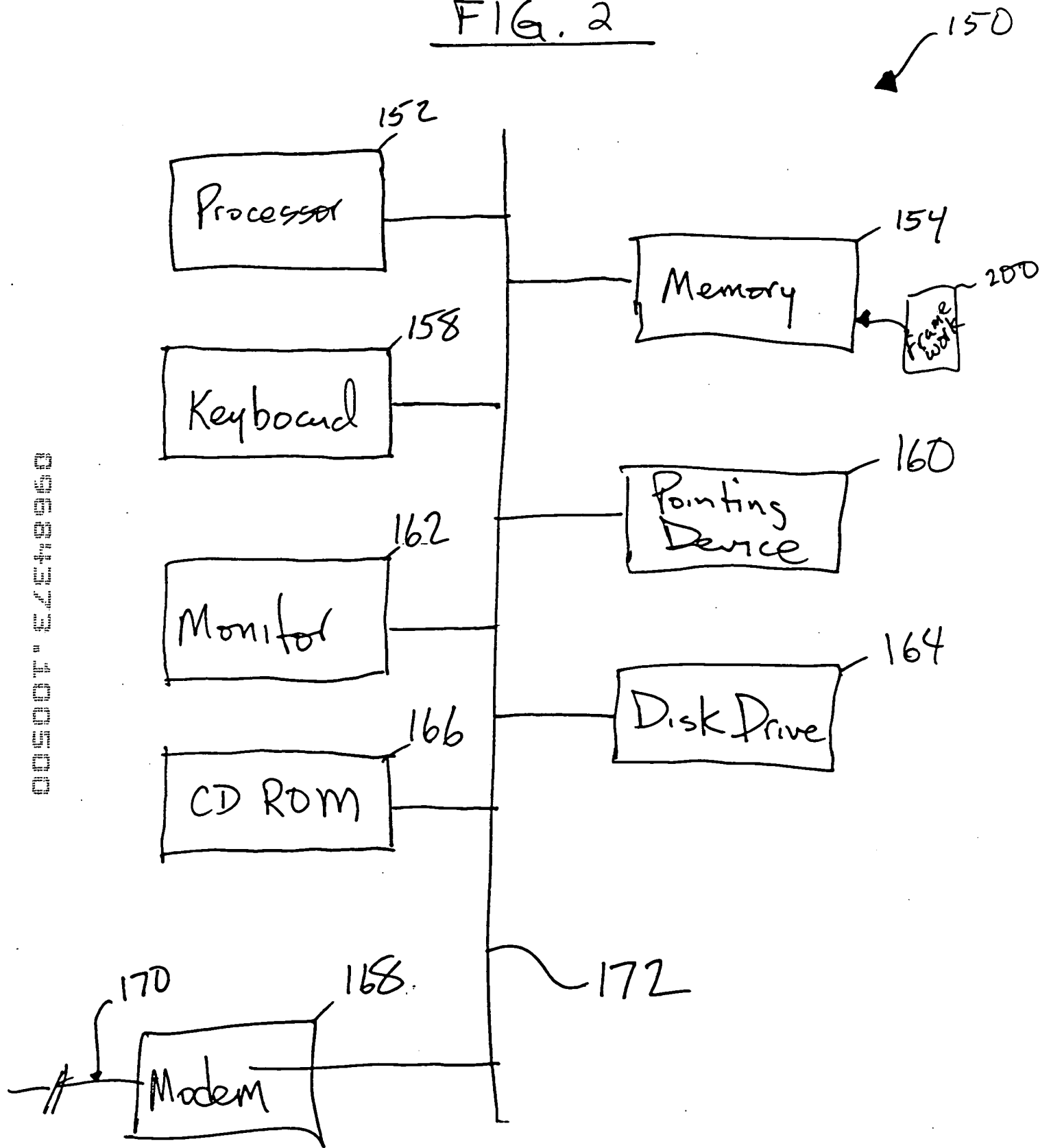*102-1*

Remote User 2

102-2-

Figure 1: Virtual Network Environments

The above diagram illustrates the virtual network environments. Virtual Network Environment 1 contains one application WEB SITE 1, is defined by the Virtual Network Address 10.10.0.1 resides solely on computer 1. Virtual Network Environment 2 contains two applications, WEB SITE 2 and APP 2, and spans two computers, Computer 2 and Computer 3. The virtual network of VNE 2 is 10.10.2.0.
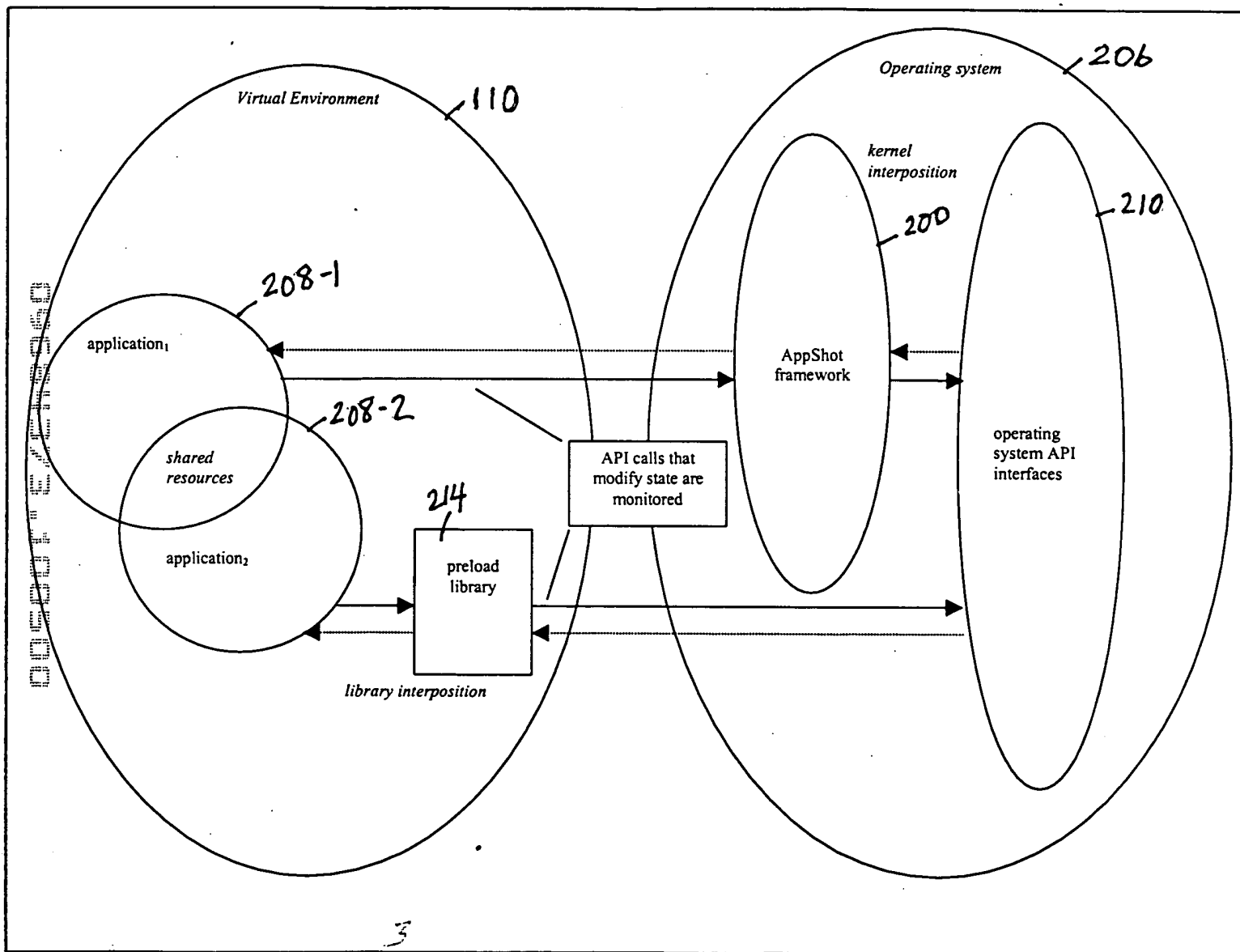
FIG. 1

FIG. 2

150

Processor 152

Memory 154

Frame Work 200

Keyboard 158

Pointing Device 160

Monitor 162

Disk Drive 164

CD ROM 166

170

168

172

Modem

Virtual Environment 110

208-1

application₁

208-2

shared resources

application₂

214

preload library

library interposition

API calls that modify state are monitored

Operating system 206

kernel interposition

200

AppShot framework

210

operating system API interfaces

Figure 1. Tracking application state via library and kernel interposition.

FIG 3

Virtual Environment ~110

application₁ ~208-1

shared resources

208-2

application₂

preload library ~214

Operating system ~206

operating system API interfaces ~210

~200

AppShot framework

216

process management subsystem

218 snapshot driver

Restore Driver

220

image of process saved

~164

Storage

(a) (b) (c) (d)

Process state is retrieved using several different methods:

a) via the interposed preload library
b) process management interfaces
c) AppShot framework
d) standard API calls

Figure 2. Capture of an application's run-time state.

FIG. 4

failed _254_

Request snapshot _250_

pass in AID

NO ~~not found~~

~~lookup~~ AID ~~found?~~ **found?** _252_

YES ~~found a_1~~

(all processes now suspended)

find a process with AID that matches a_1 ? _256_

NO ~~not found (no more processes)~~

yes ~~found p_1~~

(check for more processes)

suspend process _258_

Is state consistent and are threads quiesced? _260_

yes

no

(try again later)

resume any suspended processes _262_

Remove registered global state (i.e. semaphores) _278_

terminate process _280_

More processes? _282_

yes

no

done

no

More processes? _276_

yes

(next process)

For each process... _264_

For each type of state... _266_

(next state)

Is this state virtualized? _264_

no

yes

Retrieve virtualized state _266_

Retrieve state _268_

State changed since last snapshot? _270_

no

yes

dump state delta _272_

More state? _274_

yes

no

Figure 3. Flowchart of snapshot.

FIG. 5

failed

Restore ——300

AID

not found

lookup
AID
found? ——302

found a_1

find snapshot
image with
matching AID ——304

(check for more
images)

found i

no more
processes

find global/shared
state in i ——306

Recreate global or
shared state ——308

For each image…

Create process
(inheriting global/shared
state) ——310

isolate process to prevent
inter-process state changes ——312

For each type of state…

Reset state ——314

Is this state
virtualized? ——316

no

yes

Bind system state
to virtual defintion. ——318

(next
image)

yes

More
images? ——328

no

(next
state)

Put this process
in synchronized
wait. ——326

More
state? ——324

yes

no

Remove traces and state
induced during restore of
the process ——330

Shared state with
another process?
(i.e. pipe) ——320

no

yes

reconnect shared state
to other process ——322

Synchronized resume of
all processes ——332

done

Figure 4. Flowchart of restore. FIG 6

Resource Name Size	
Resource Descriptor Size	
Resource Type	
Resource Name	
Resource Data	

Alarm Info	Semaphore Info
File Info	Platform Name
Signal Info	Data Queued Info
Current directory	Process Status
Application Virt ID	Credentials Info
Application Virt Net Mask	File Locking Info
Dynamic Symbolic Link Info	MMap Memory Info
Resource limit Info	Process Map Info
Process Info	Schedule control info
Snapshot Info	Lightweight Process info

FIG. 7
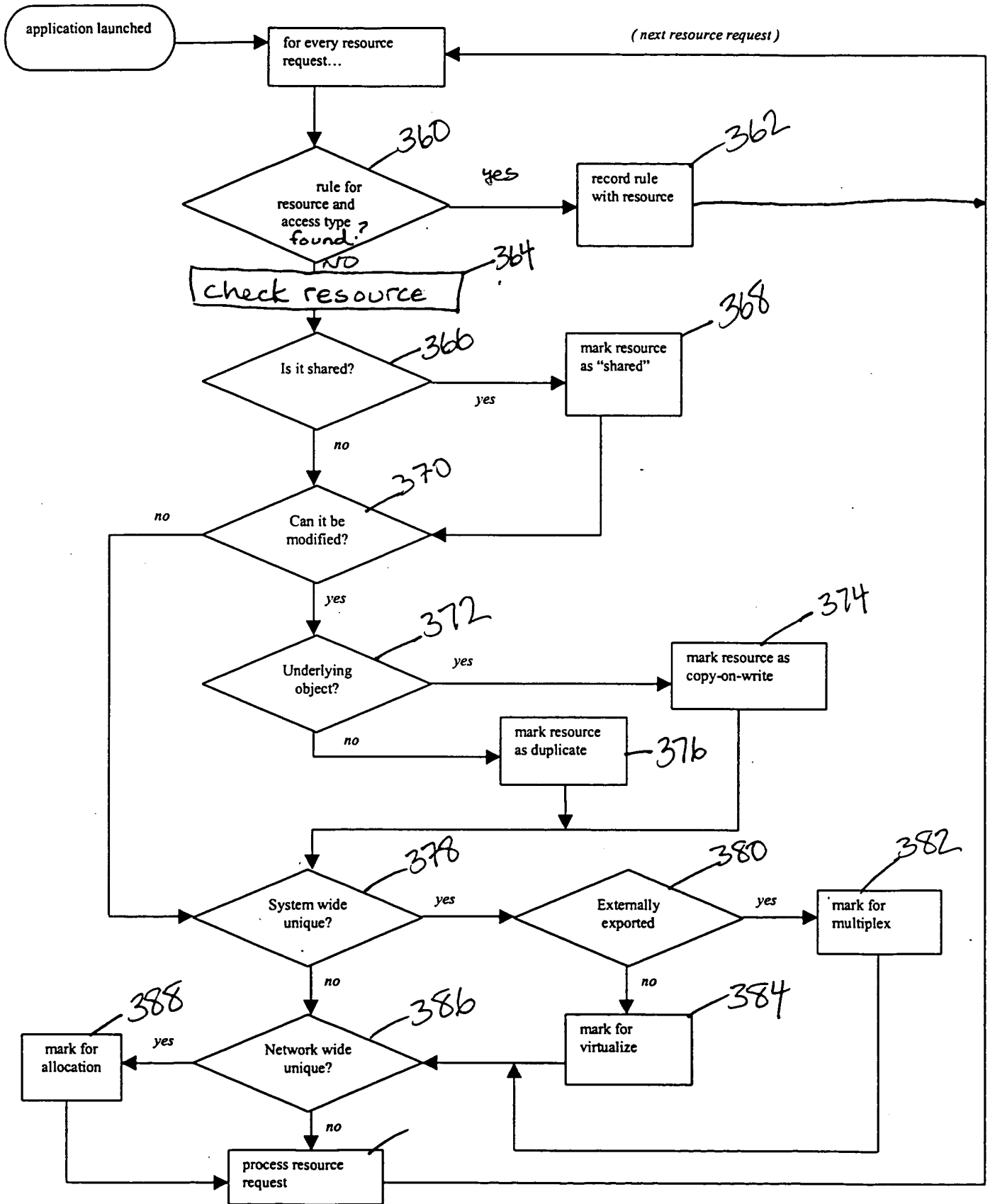
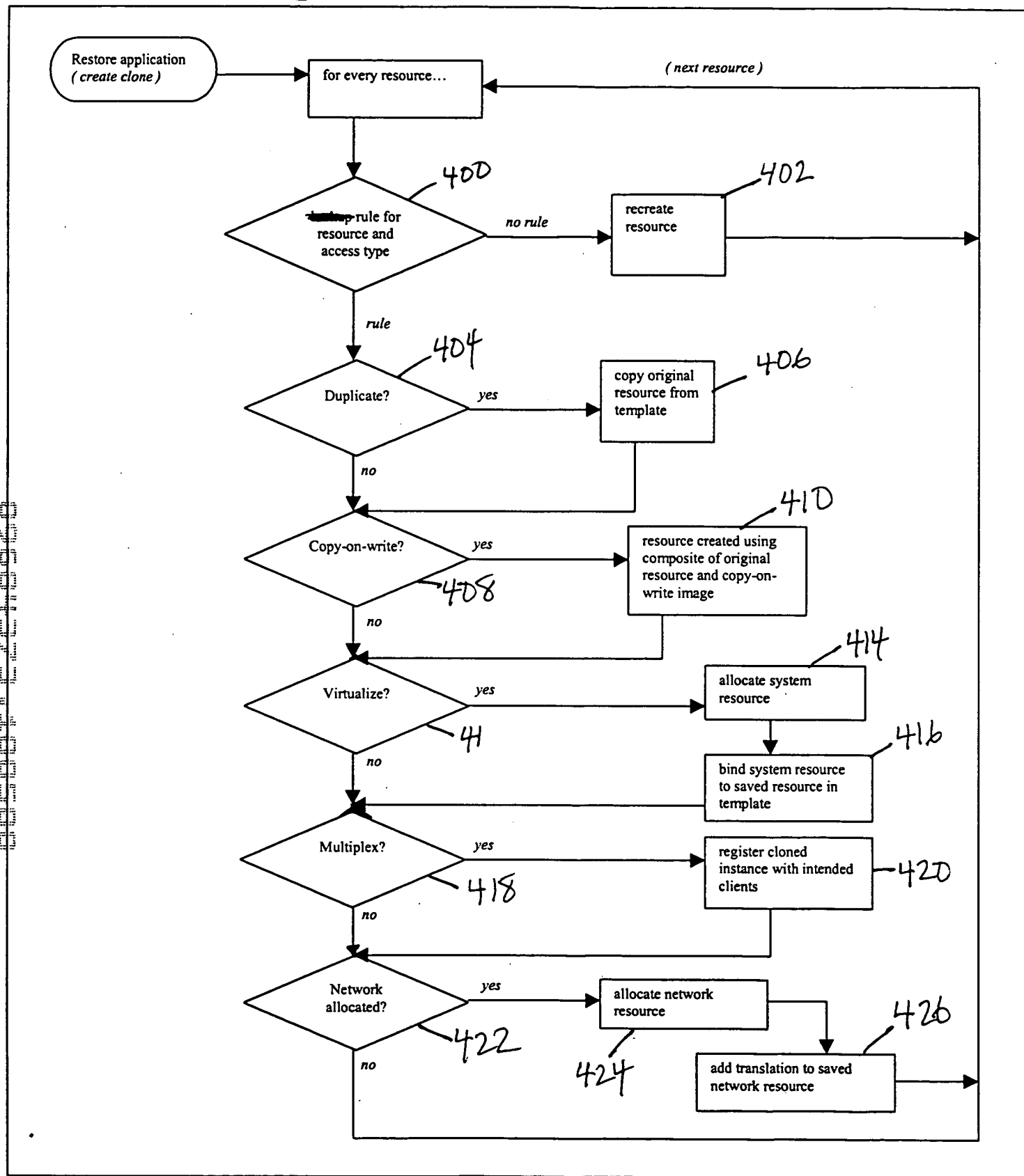*Figure 1. Flowchart of creation of virtual template.* FIG.8

Restore application
(create clone)

for every resource...

(next resource)

400 rule for resource and access type

no rule → recreate resource **402**

rule

404 Duplicate?

yes → copy original resource from template **406**

no

Copy-on-write?

yes → resource created using composite of original resource and copy-on-write image **410**

408

no

Virtualize?

yes → allocate system resource **414**

→ bind system resource to saved resource in template **416**

41

no

Multiplex?

yes → register cloned instance with intended clients **420**

418

no

Network allocated?

yes → allocate network resource **424**

→ add translation to saved network resource **426**

422

no

Figure 2. Cloning of a virtual template. FIG. 9

500

206

Operating system kernel

AppShot harness
($process_x$)

(510)

AppShot framework

200

(512)

(516)

a_1 virtual resource translation tables

502

semaphores   process ID   shared memory

208

application
($process_y$)

(514)

$process_y$ state

$process_x$ state

5F8
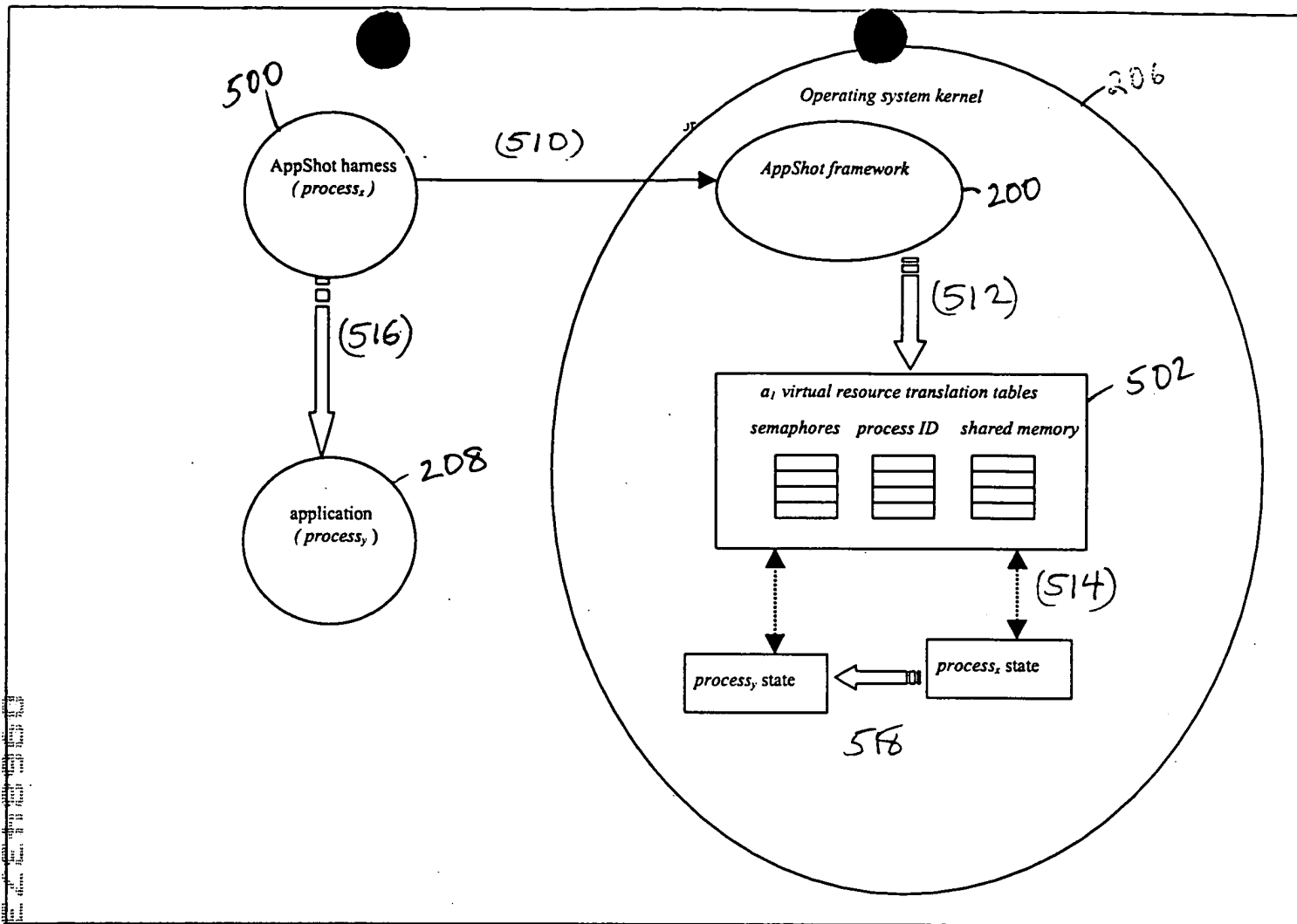
Figure 1. Registration of an application using virtual resource ID's.

FIG. 10
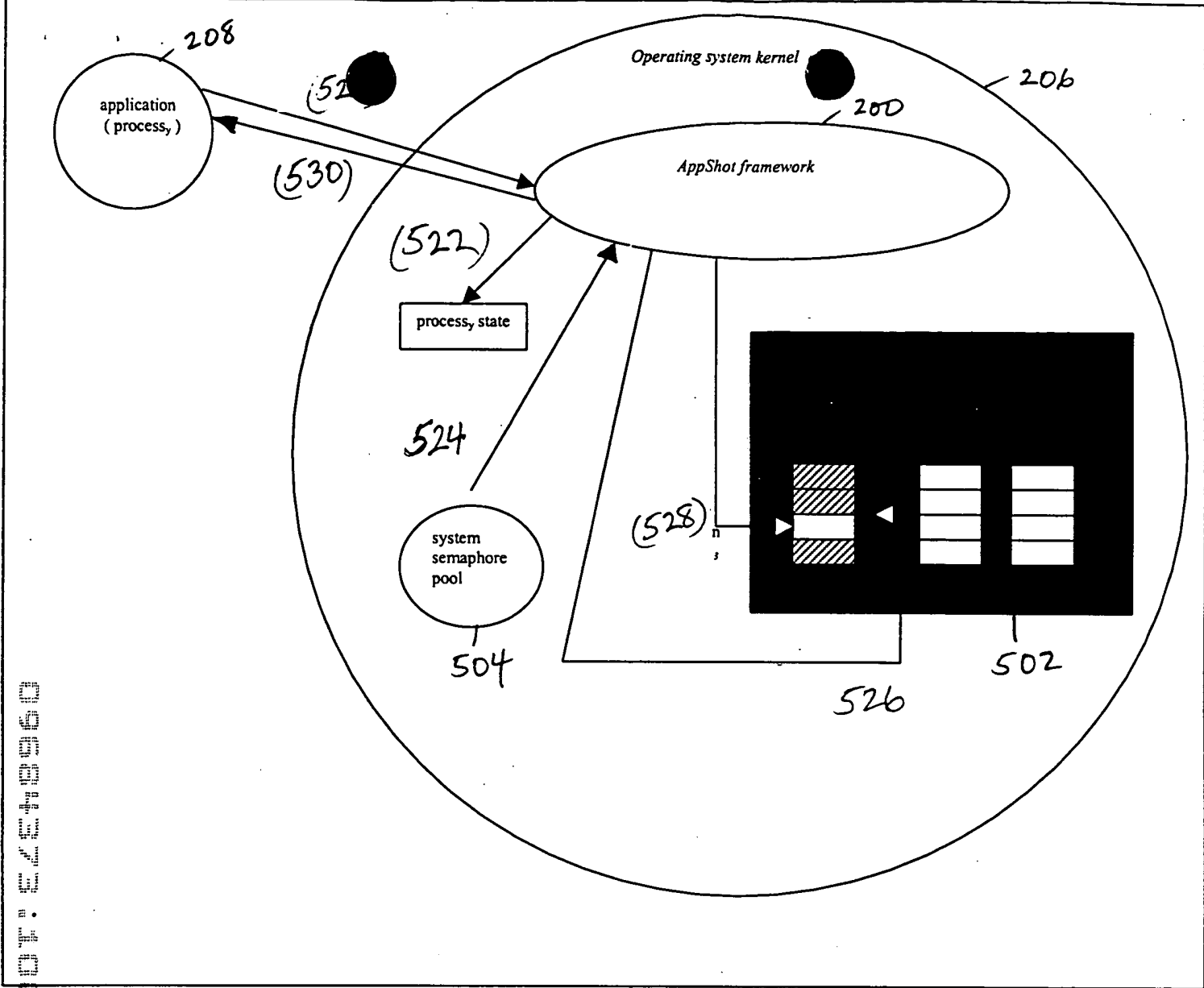
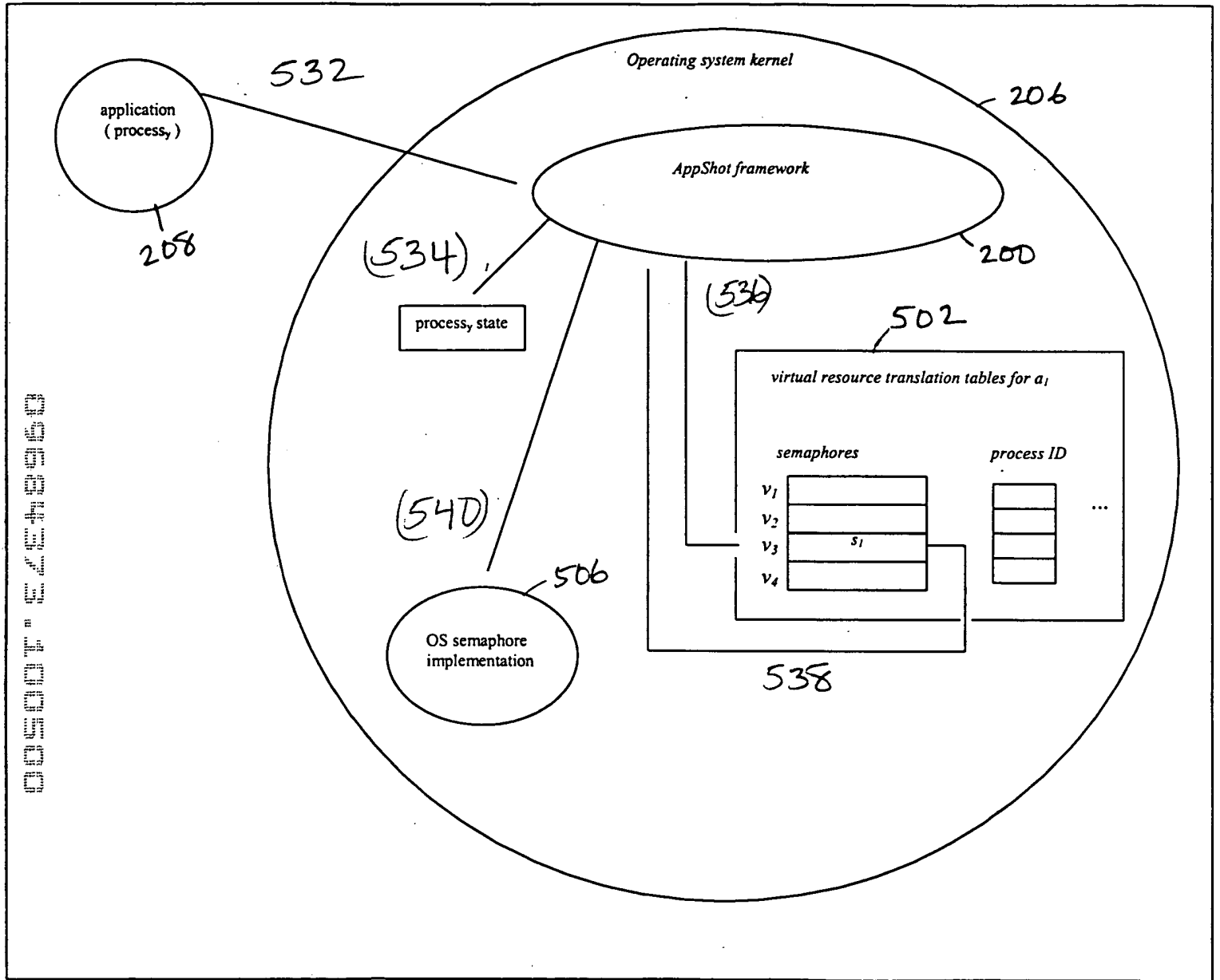*Figure 2. Allocation of a virtual resource (i.e. semaphore)*

FIG. 11

Figure 3. Translation of a virtual resource (i.e. semaphore), virtual to system

FIG. 12

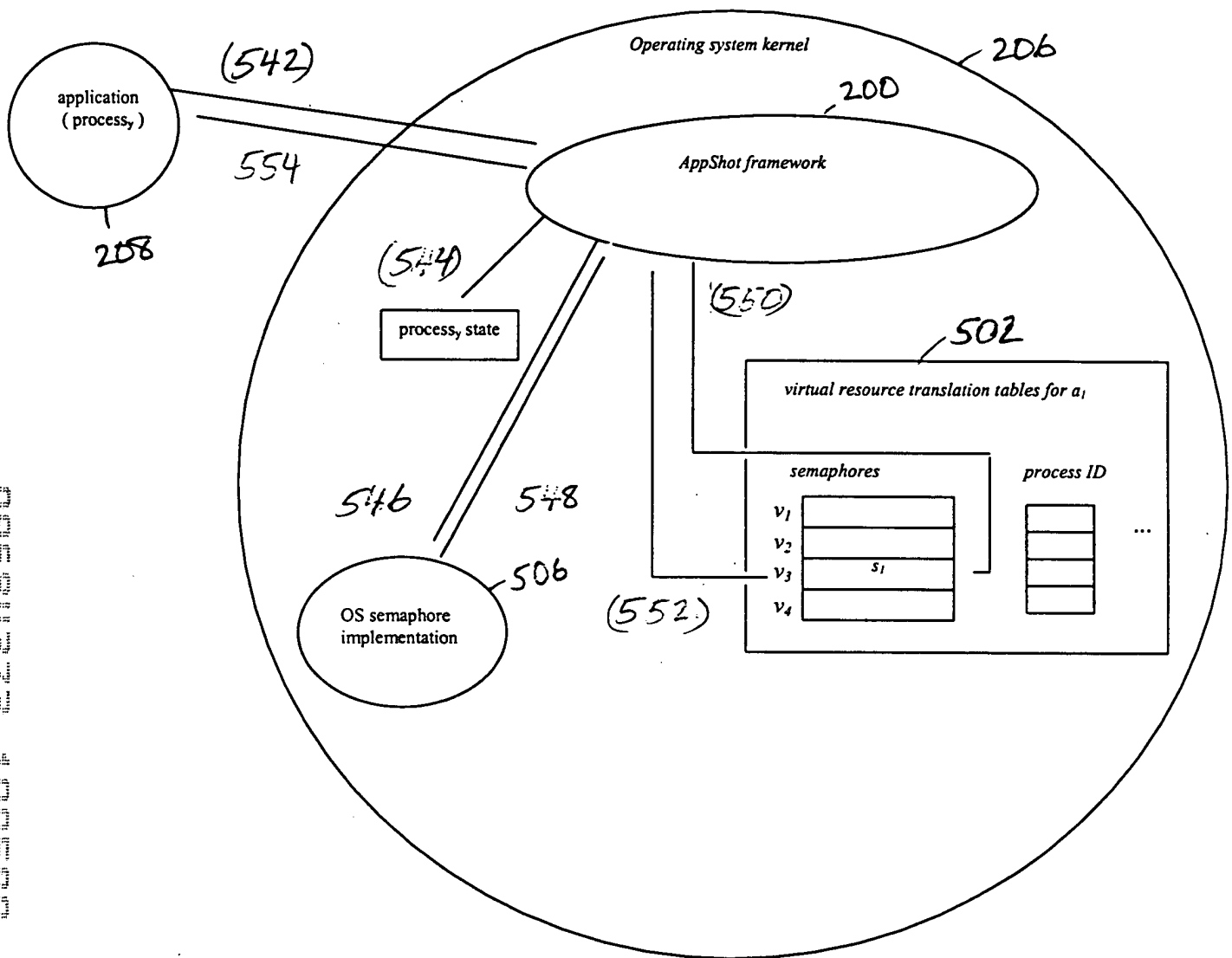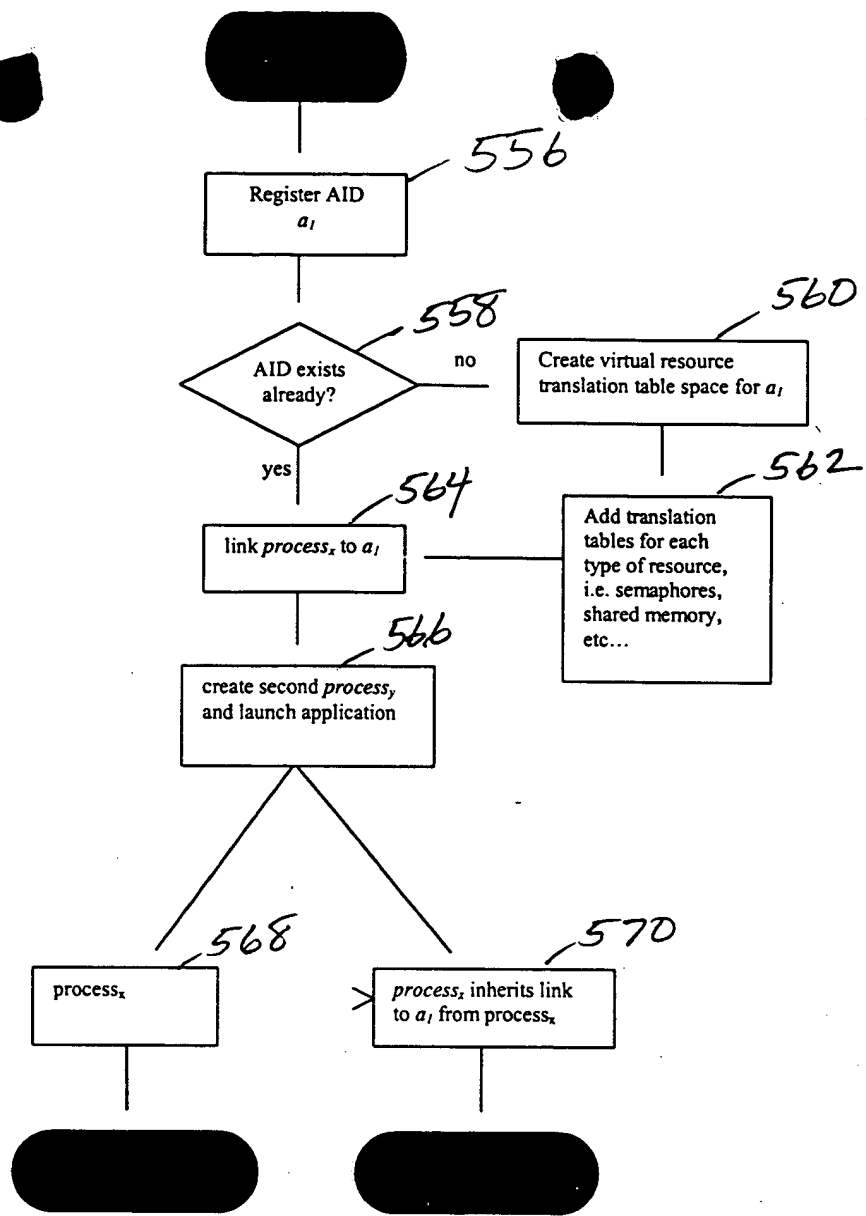Figure 4. Reverse translation of a virtual resource (i.e. semaphore), system to virtual

FIG. 13

Register AID
a_l

556

AID exists
already?

no

558

Create virtual resource
translation table space for a_l

560

yes

link $process_x$ to a_l

564

Add translation
tables for each
type of resource,
i.e. semaphores,
shared memory,
etc...

562

create second $process_y$
and launch application

566

$process_x$

568

$process_x$ inherits link
to a_l from $process_x$

570

Figure 5. Flowchart describing creation of virtual translation tables.

FIG. 14

FIG. 15

580 lookup process AID?

not found

found (a_1)

582 Interface uses semaphore ID as parameter?

no

yes (v_1)

584 lookup translation for $a_1 : v_1$

not found

found (s_1)

586 substitute s_1 for v_1

588 operating system semaphore interface

590 Was semaphore allocated?

no

yes (s_2)

592 Does interface return semaphore?

no

yes (s_3)

598 allocate virtual semaphore

(v_2)

594 reverse lookup translation for $a_1 : s_3$

(v_3)

600 insert translation for $v_2 . a_1 : s_2$

596 substitute v_3 for s_3

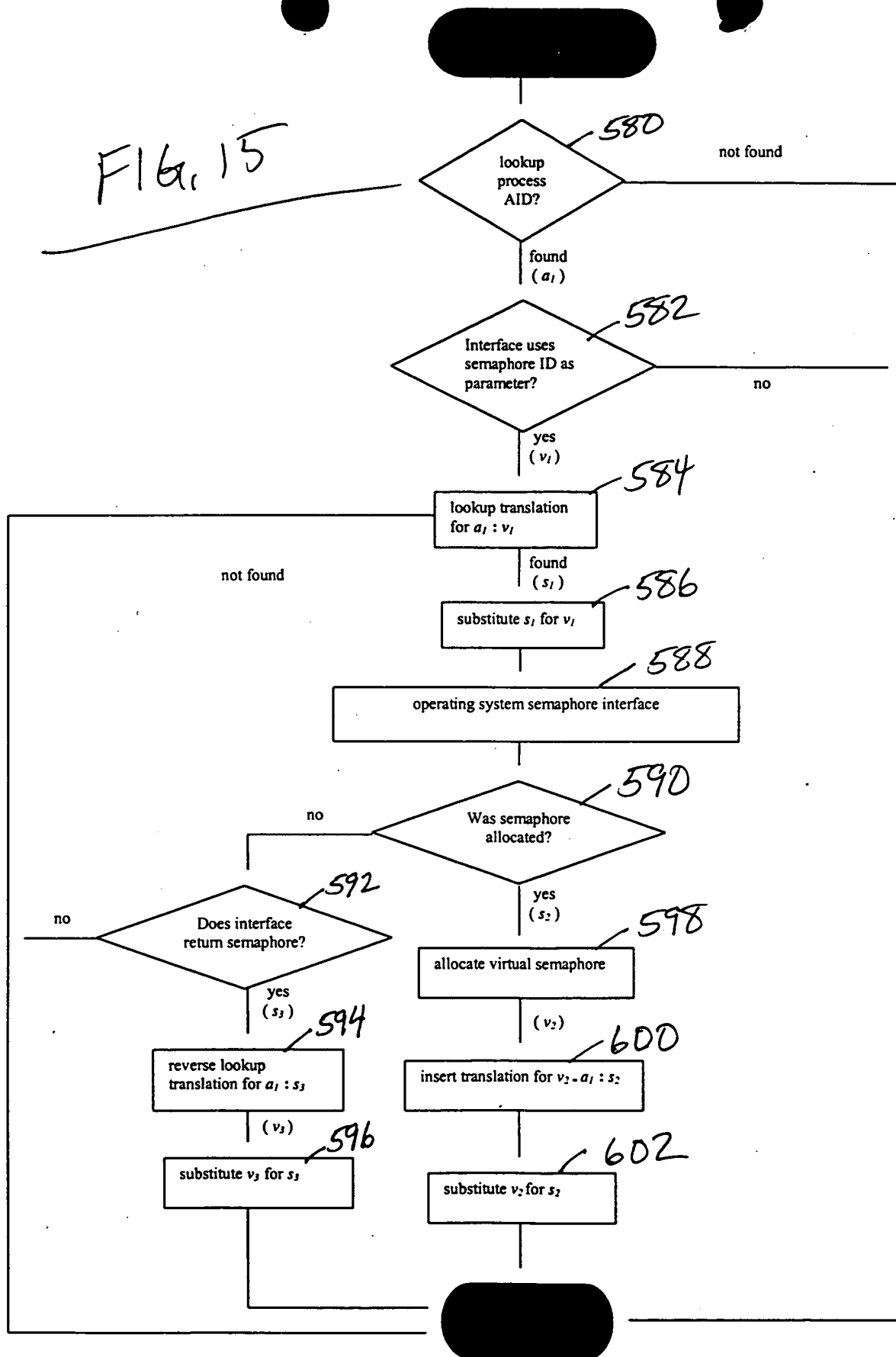**602** substitute v_2 for s_2

Figure 6. Flowchart depicting translation of a virtual resource (i.e. semaphore)